MANUAL TÉCNICO

PROYECTO #2

USAC
TRICENTENARIA
Universidad de San Carlos de Guatemala

FIUSAC
Universidad de San Carlos de Guatemala

Organización de Lenguajes y de Compiladores 1

Javier Alejandro Gutierrez de León

202004765

Guatemala, 29 de abril de 2022

Introducción

Se solicita por parte de la Escuela de Ciencias y Sistemas de la Facultad de Ingeniería, crear un lenguaje de programación para que los estudiantes de Introducción a la Programación y Computación 1, aprendan a programar y tener conocimiento de todas las generalidades de un lenguaje de programación, ya que este lenguaje será utilizado para generar sus primeras prácticas de laboratorio del curso antes mencionado. Por lo que se debe crear el proyecto llamado CompScript, el cual es un lenguaje interpretado que acpta archivos con extensión ***".cst"*** del cual se realizarán los análisis léxico, sintáctico y semántico, además de ejecutar todas las sentencias. Luego de esto se genera Reporte de Errores, en el cual se mostrarán todos los errores encontrados al realizar el análisis léxico, sintáctico y semántico. Además se generará un Árbol AST (Árbol de Análisis Sintáctico) que se debe generar una imagen del árbol de análisis sintáctico que se genera al realizar los análisis. Y por último se genra un Reporte de Tabla de Símbolos, donde se muestrán todas las variables, métodos y funciones que han sido declarados dentro del flujo del programa.

Objetivos

Objetivos Generales

- ✔ Aplicar los conocimientos sobre la fase de análisis léxico y sintáctico de un compilador para la realización de un intérprete sencillo, con las funcionalidades principales para que sea funcional.

Objetivos Específicos

- ✔ Reforzar los conocimientos de análisis léxico y sintáctico para la creación de un lenguaje de programación.
- ✔ Aplicar los conceptos de compiladores para implementar el proceso de interpretación de código de alto nivel.
- ✔ Aplicar los conceptos de compiladores para analizar un lenguaje de programación y producir las salidas esperadas.
- ✔ Aplicar la teoría de compiladores para la creación de soluciones de software.
- ✔ Aplicar conceptos de contenedores para generar aplicaciones livianas.
- ✔ Generar aplicaciones utilizando arquitecturas Cliente-Servidor.

Especificación Técnica

Requisitos de Hardware

- RAM: 128 MB
- Procesador: Mínimo Pentium 2 a 266 MHz
- Espacio en disco: 124 MB para JRE; 2 MB para Java Update; 45MB para programa fuente

Requisitos de software

- **Sistema operativo**

  - Windows 10 (8u51 y superiores)
  - Windows 8.x (escritorio)
  - Windows 7 SP1
  - Windows Vista SP2
  - Windows Server 2008 R2 SP1 (64 bits)
  - Windows Server 2012 y 2012 R2 (64 bits)
  - Oracle Linux 5.5+1
  - Oracle Linux 6.x (32 bits), 6.x (64 bits)2
  - Oracle Linux 7.x (64 bits)2 (8u20 y superiores)
  - Red Hat Enterprise Linux 5.5+1 6.x (32 bits), 6.x (64 bits)2
  - Red Hat Enterprise Linux 7.x (64 bits)2 (8u20 y superiores)
  - Suse Linux Enterprise Server 10 SP2+, 11.x
  - Suse Linux Enterprise Server 12.x (64 bits)2 (8u31 y superiores)
  - Ubuntu Linux 12.04 LTS, 13.x
  - Ubuntu Linux 14.x (8u25 y superiores)
  - Ubuntu Linux 15.04 (8u45 y superiores)
  - Ubuntu Linux 15.10 (8u65 y superiores)
  - Mac con Intel que ejecuta Mac OS X 10.8.3+, 10.9+

- **Exploradores (Para visualizar reporte de errres):**
  - Google Chrome (en todas las plataformas)
  - Mozilla Firefox (en todas las plataformas)
  - Internet Explorer (Windows)
  - Microsoft Edge (Windows, Android, iOS, Windows 10 móvil)
  - Safari (Mac, iOS)
  - Opera (Mac, Windows)

- **Tecnologías utilizadas**
  - **Node JS (16.14.0):** es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor basado en el lenguaje de programación JavaScript,.
  - **Angular (13.0.3):** es un framework para aplicaciones web desarrollado en TypeScript, de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página.
  - **HTML5:** lenguaje de marcado para la elaboración de páginas web, con el que se realizaron los reportes de errores lexicos y sintacticos.
  - **Boostrap:** Librería para el diseño del reporte HTML.
  - **Graphviz:** Es un conjunto de herramientas de software para el diseño de diagramas definido en el lenguaje descriptivo DOT.
  - **Javascript:** es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Creacion del analisis Léxico

Tabla de tokens

Token	Descripción	Expresión regular
Se ignora	Espacios en blanco	[\r\t\n\s]+
Se ignora	Comentario simple	[/][/].*
Se ignora	Comentario múltiple	[/][*][^*]*[*]+([^/*][^*]*[*]+)*[/]
R_INT	Palabra reservada para el tipo de dato entero	“int”
R_DOUBLE	Palabra reservada para el tipo de dato decimal	“double”
R_BOOLEAN	Palabra reservada para el tipo de dato booleano	“boolean”
R_CHAR	Palabra reservada para el tipo de dato caracter	“char”
R_STRING	Palabra reservada para el tipo de dato string	“string”
R_TRUE	Palabra reservada true	"true"
R_FALSE	Palabra reservada false	"false"
R_NEW	Palabra reservada new	"new"
R_IF	Palabra reservada if	“if”
R_ELSE	Palabra reservada else	“else”
R_SWITCH	Palabra reservada switch	“switch”
R_CASE	Palabra reservada case	“case”
R_DEFAULT	Palabra reservada default	“default”
R_WHILE	Palabra reservada while	“while”
R_FOR	Palabra reservada for	“for”
R_DO	Palabra reservada do	“do”
R_BREAK	Palabra reservada break	“break”
R_CONTINUE	Palabra reservada continue	“continue”
R_RETURN	Palabra reservada return	“return”
R_VOID	Palabra reservada void	“void”
R_PRINT	Palabra reservada print	“print”
R_PRINTLN	Palabra reservada println	“println”
R_TOLOWER	Palabra reservada tolower	“tolower”
R_TOUPPER	Palabra reservada toupper	“toupper”
R_ROUND	Palabra reservada round	“round”
R_LENGTH	Palabra reservada length	“length”
R_TYPEOF	Palabra reservada typeof	“typeof”
R_TOSTRING	Palabra reservada tostring	“tostring”
R_TOCHAR ARRAY	Palabra reservada tochararray	“tochararray”
R_RUN	Palabra reservada run	“run”

DOSPTS	Carácter dos puntos	“:”
PTCOMA	Carácter punto y coma	“;”
LLAVIZQ	Carácter llave apertura	“{”
LLAVDER	Carácter llave cierre	“}”
PARIZQ	Carácter paréntesis apertura	“(”
PARDER	Carácter paréntesis cierre	“)”
CORIZQ	Carácter corchete apertura	“[”
CORDER	Carácter corchete cierre	“]”
COMA	Carácter coma	“,”
IGUALACION	Carácter igual que	“==”
DIFERENCIACION	Carácter diferente que	“!=”
MENIGUALQ	Carácter menor o igual que	“<=”
MAYIGUALQ	Carácter mayor o igual que	“=>”
MENORQ	Carácter menor que	“<”
MAYORQ	Carácter mayor que	“>”
DIF	Carácter not	“!”
OR	Caracteres or	“||”
AND	Caracteres and	“&&”
INTERROG	Carácter interrogación	“?”
INCRE	Caracteres incremento	“++”
DECRE	Caracteres decremento	“- -“
MAS	Carácter mas	“+”
MENOS	Carácter menos	“-”
POR	Carácter por	“*”
DIV	Carácter dividido	“/”
POTENCIA	Carácter potencia	“^”
MODULO	Carácter modulo	“%”
IGUAL	Carácter igual	“=”
DECIMAL	Numero decimal	[0-9]+("."[0-9]+)\b
ENTERO	Numero entero	[0-9]+\b
IDENTIFICADOR	Identificador	([a-zA-Z])[a-zA-Z0-9_]*
CARACTER	Único carácter entre comillas simples	\' ((\\[\'\"\\bfnrtv])|([^\"\\])) \'
CADENA	Cadena entre comillas dobles	\" ((\\[\'\"\\bfnrtv])|([^\"\\]+))*\"

Gramática Libre de Contexto

G=(V,T,P,S)

V Conjunto de variables

ini, instrucciones, instrucción, tipo, expresion, lista, listavec, declaracion, dec, asig, asig_solo, casteo, inc_dec, dec_vectores, acs_vectores, mod_vectores, terna, sen_if, sen_switch, list_case, s_case, s_default, sen_while, sen_for, sen_dowhile, sen_return, funcion, parametros, parmetro, metodos, llamada, parametros_llamada, fprint, fprintln, ftolower, ftoupper, fround, flength, ftypeof, ftostring, ftochararray, frun.

T Conjunto de terminales

R_INT, R_DOUBLE, R_BOOLEAN, R_CHAR, R_STRING, R_TRUE, R_FALSE, R_NEW, R_IF, R_ELSE, R_SWITCH, R_CASE, R_DEFAULT, R_WHILE, R_FOR, R_DO, R_BREAK, R_CONTINUE, R_RETURN, R_VOID, R_PRINT, R_PRINTLN, R_TOLOWER, R_TOUPPER, R_ROUND, R_LENGTH, R_TYPEOF, R_TOSTRING, R_TOCHARARRAY, R_RUN, DOSPTS, PTCOMA, LLAVIZQ, LLAVDER, PARIZQ, PARDER, CORIZQ, CORDER, COMA, IGUALACION, DIFERENCIACION, MENIGUALQ, MAYIGUALQ, MENORQ, MAYORQ, DIF, OR, AND, INTERROG, INCRE, DECRE, MAS, MENOS, POR, DIV, POTENCIA, MODULO, IGUAL, DECIMAL, ENTERO, IDENTIFICADOR, CARÁCTER, CADENA,

P Conjunto finito de producciones

<ini> ::= <instrucciones> EOF
| EOF

<instrucciones> ::= <instrucciones> <instruccion>
| <instruccion>

<instruccion> ::= <declaracion> PTCOMA
| <inc_dec> PTCOMA
| <dec_vectores> PTCOMA
| <mod_vectores> PTCOMA
| <sen_if>

| <asig_solo> PTCOMA
| <sen_switch>
| <sen_while>
| <sen_for>
| <sen_dowhile> PTCOMA
| <sen_return> PTCOMA
| R_BREAK PTCOMA
| R_CONTINUE PTCOMA
| <metodos>
| <funcion>
| <llamada> PTCOMA
| <fprint> PTCOMA
| <fprintln> PTCOMA
| <frun> PTCOMA

<tipo> ::= R_INT
| R_DOUBLE
| R_BOOLEAN
| R_CHAR
| R_STRING

<expresion> ::= <expresion> POTENCIA <expresion>
| <expresion> MAS <expresion>
| <expresion> MENOS <expresion>
| <expresion> POR <expresion>
| <expresion> DIV <expresion>
| <expresion> MODULO <expresion>
| MENOS <expresion>
| PARIZQ <expresion> PARDER
| DIF <expresion>
| <expresion> IGUALACION <expresion>
| <expresion> DIFERENCIACION <expresion>
| <expresion> MENIGUALQ <expresion>
| <expresion> MAYIGUALQ <expresion>
| <expresion MENORQ <expresion>
| <expresion MAYORQ <expresion>

| <expresion> OR <expresion>
| <expresion> AND <expresion>
| ENTERO
| DECIMAL
| R_TRUE
| R_FALSE
| IDENTIFICADOR
| CARACTER
| CADENA
| <acs_vectores>
| <casteo>
| <ftolower>
| <ftoupper>
| <fround>
| <flength>
| <ftypeof>
| <ftostring>
| <ftochararray>
| <llamada>
| <inc_dec>
| <terna>

<lista> ::= <lista> COMA <expresion>
| <expresion>

<listavec>::= <listavec> COMA CORIZQ <lista> CORDER
| CORIZQ <lista> CORDER

<declaracion> ::= <tipo> <dec>

<dec> ::= <dec> COMA <asig>
| <asig >

<asig> ::= IDENTIFICADOR IGUAL <expresion>
| IDENTIFICADOR

<asig_solo> ::= IDENTIFICADOR IGUAL <expresion>

<casteo> ::= PARIZQ <tipo> PARDER <expresion>

<inc_dec> ::= <expresion> INCRE
| <expresion> DECRE

<dec_vectores> ::= <tipo> IDENTIFICADOR CORIZQ CORDER IGUAL R_NEW <tipo> CORIZQ <expresion> CORDER
| <tipo> IDENTIFICADOR CORIZQ CORDER CORIZQ CORDER IGUAL R_NEW <tipo> CORIZQ <expresion> CORDER CORIZQ <expresion> CORDER
| <tipo> IDENTIFICADOR CORIZQ CORDER IGUAL CORIZQ <lista> CORDER
| <tipo> IDENTIFICADOR CORIZQ CORDER CORIZQ CORDER IGUAL CORIZQ <listavec> CORDER
| <tipo> IDENTIFICADOR CORIZQ CORDER IGUAL <expresion>
| <tipo> IDENTIFICADOR CORIZQ CORDER CORIZQ CORDER IGUAL CORIZQ <lista> CORDER

<acs_vectores> ::= IDENTIFICADOR CORIZQ <expresion> CORDER
| IDENTIFICADOR CORIZQ <expresion> CORDER CORIZQ <expresion> CORDER

<mod_vectores> ::= IDENTIFICADOR CORIZQ <expresion> CORDER IGUAL <expresion>
| IDENTIFICADOR CORIZQ <expresion> CORDER CORIZQ <expresion> CORDER IGUAL <expresion>

<terna> ::= <expresion> INTERROG <expresion> DOSPTS <expresion>

<sen_if> ::= R_IF PARIZQ <expresion> PARDER LLAVIZQ <instrucciones> LLAVDER
| R_IF PARIZQ <expresion> PARDER LLAVIZQ <instrucciones> LLAVDER R_ELSE LLAVIZQ <instrucciones> LLAVDER
| R_IF PARIZQ <expresion> PARDER LLAVIZQ <instrucciones> LLAVDER R_ELSE <sen_if>

sen_switch ::= R_SWITCH PARIZQ <expresion> PARDER LLAVIZQ <list_case> <s_default> LLAVDER
| R_SWITCH PARIZQ <expresion> PARDER LLAVIZQ <list_case> LLAVDER
| R_SWITCH PARIZQ <expresion> PARDER LLAVIZQ <s_default> LLAVDER

<list_case> ::= <list_case> <s_case>
|<s_case>

<s_case> ::= R_CASE <expresion> DOSPTS <instrucciones>

<s_default> ::= R_DEFAULT DOSPTS <instrucciones>

<sen_while> ::= R_WHILE PARIZQ <expresion> PARDER LLAVIZQ <instrucciones> LLAVDER

<sen_for> ::= R_FOR PARIZQ <declaracion> PTCOMA <expresion> PTCOMA <inc_dec> PARDER LLAVIZQ <instrucciones> LLAVDER
| R_FOR PARIZQ <asig_solo> PTCOMA <expresion> PTCOMA <inc_dec> PARDER LLAVIZQ <instrucciones> LLAVDER
| R_FOR PARIZQ <declaracion> PTCOMA <expresion> PTCOMA <asig_solo> PARDER LLAVIZQ <instrucciones> LLAVDER
| R_FOR PARIZQ <asig_solo> PTCOMA <expresion> PTCOMA <asig_solo> PARDER LLAVIZQ <instrucciones> LLAVDER

<sen_dowhile> ::= R_DO LLAVIZQ <instrucciones> LLAVDER R_WHILE PARIZQ <expresion> PARDER

<sen_return> ::= R_RETURN <expresion>

<funcion> ::= IDENTIFICADOR PARIZQ <parametros> PARDER DOSPTS <tipo> LLAVIZQ <instrucciones> LLAVDER
| IDENTIFICADOR PARIZQ PARDER DOSPTS <tipo> LLAVIZQ <instrucciones> LLAVDER

<parametros> ::= <parametros> COMA <parmetro>
| <parmetro>

<parmetro> ::= <tipo> IDENTIFICADOR

<metodos> ::= IDENTIFICADOR PARIZQ <parametros> PARDER LLAVIZQ <instrucciones> LLAVDER

| IDENTIFICADOR PARIZQ <parametros> PARDER DOSPTS R_VOID LLAVIZQ <instrucciones> LLAVDER
| IDENTIFICADOR PARIZQ PARDER LLAVIZQ <instrucciones> LLAVDER
| IDENTIFICADOR PARIZQ PARDER DOSPTS R_VOID LLAVIZQ <instrucciones> LLAVDER

<llamada> ::= IDENTIFICADOR PARIZQ PARDER
| IDENTIFICADOR PARIZQ <parametros_llamada> PARDER

<parametros_llamada> ::= <parametros_llamada> COMA <expresion>
| <expresion>

<fprint> ::= R_PRINT PARIZQ <expresion> PARDER

<fprintln> ::= R_PRINTLN PARIZQ <expresion> PARDER

<ftolower> ::= R_TOLOWER PARIZQ <expresion> PARDER

<ftoupper> ::= R_TOUPPER PARIZQ <expresion> PARDER

<fround> ::= R_ROUND PARIZQ <expresion> PARDER

<flength> ::= R_LENGTH PARIZQ <expresion> PARDER

<ftypeof> ::= R_TYPEOF PARIZQ <expresion> PARDER

<ftostring> ::= R_TOSTRING PARIZQ <expresion> PARDER

<ftochararray> ::= R_TOCHARARRAY PARIZQ <expresion> PARDER

<frun> ::= R_RUN <llamada>

S Simbolo Inicial

Ini

Precedencia

Por la izquierda 'INTERROG' 'DOSPTS'
Por la izquierda 'OR'
Por la izquierda 'AND'
Por la derecha 'DIF'
Por la izquierda 'IGUALACION' 'DIFERENCIACION' 'MENORQ' 'MENIGUALQ' 'MAYORQ' 'MAYIGUALQ'

Por la izquierda 'MAS' 'MENOS'
Por la izquierda 'DIV' 'POR' 'MODULO'
No asociativa 'POTENCIA'
Por la derechaUMINUS
Por la izquierda 'PARIZQ' 'PARDER'
Por la derecha 'INCRE' 'DECRE'

Lógica del programa (Métodos principales)

Nombre	Descripción	Parámetros
parser.parse(codigo)	Función con la cual se ejecuta el analisis Lexico y Sintactico	• **Codigo:** Codigo obtenido del archivo cst
AST.imprimir(raiz)	Función para obtener el dot para generar el arbol.	• **Raiz:** Raiz del AST generado en el proceso del analisis lexico y sintactico
semantico.interpretar(raiz, ambito, lugar, pasada)	Función para ejecutar el analisis semantico	• **Raiz:** Raiz del AST generado en el proceso del analisis lexico y sintactico • **Ambito:** entorno en que se esta analizando • **Lugar:** tipo de sentencia que esta analizando • **Pasada:** Valor booleano para indicar si es primera pasada o no
funFor(cond,actualizacion,raiz, ambito,primero,ini,pasada)	función para analizar el codigo del ciclo for	• **cond:** condicion del ciclo for • **actualizacion:** Actualización del ciclo for • **raiz:** Raiz de las instrucciones dentro del ciclo for • **ambito:** ambito en que se encuentra • **primero:** Booleano que indica si ya se realizó la primera iteración • **ini:** Raiz del AST donde comienzan las instrucciones • **pasada:** Valor booleano para indicar si es primera pasada
evaluarLCase(exp,raiz,ambito,pasada)	Función con el que se evaluan los case del Switch	• **exp:** Expresion para entrar al case. • **Raiz:** Raiz del AST generado en el proceso del analisis lexico y sintactico • **Ambito:** entorno en que se esta analizando • **Lugar:** tipo de sentencia que esta analizando • **Pasada:** Valor booleano para indicar si es primera pasada o no

variable(tipo,raiz,ambito)	Función para asignacion o creacion de variables	• **tipo:** tipo de la variable que se declara • **raiz:** raiz del subarbol que contiene la demas informacion (asignacion o declaracion) • **ambito:** ambito en el que se declara la variable
evaluarExpresion(raiz,ambi)	Función que analiza y devuelve el resultado de operaciones aritméticas, logicas y funciones nativas que devuelven un dato	• **raiz:** subarbol que contiene la expresion que se analizará • **ambito:** ambito en que se encuentra
Tabla_simbolos	Clase que posee la estructura en la cual se ingresan los simbolos que se encuentran en la ejecución del programa	•
Errores	Clase que posee la estructura en la cual se ingresan los errores lexicos, sintacticos y semanticos	•
Fnativas	Contiene todas las funciones nativas	•
OpAritmeticas	Contine las funciones con las que se realizan las operaciones aritméticas	•
OpLogicos	Contine las funciones con las que se realizan las operaciones lógicas	•
OpRelacionales	Contine las funciones con las que se realizan las operaciones relacionales	•

Conclusiones

La utilizacion de la herramienta Jison ayudó bastante en el desarrollo de la aplicación ya que simplifican bastante el trabajo a la hora de crear el analizador lexico y sintactico ya que la cantidad de codigo creado se reduce en gran medida y el uso del AST para el analisis semantico facilita el analisis semantico en base a lo obtenido en el analisis sintactico.